ulm university universität
uulm

Institut für Angewandte Informationsverarbeitung
Dr. Andreas F. Borchert

11. November 2014
Blatt 5

Objektorientierte Programmierung mit C++ (WS 2014/2015)

Abgabe bis zum 18. November 2014, 14:00 Uhr

Lernziele:

- Entwicklung von Klassen mit dynamischen Datenstrukturen
- Tieferes Verständnis der Aufgaben eines Kopierkonstruktors, Zuweisungsoperators und Dekonstruktors.

Aufgabe 7: Knockout Geography

Hierbei handelt es sich um ein Spiel mit Worten, das auf David Silverman zurückgeht und durch Martin Gardners Kolumne in *Scientific American* bekannt wurde. Es werden typischerweise die Namen der 50 US-Bundesstaaten verwendet und es kann von beliebig vielen Spielern gespielt werden, die ringförmig angeordnet werden. Der erste Spieler kann sich eines der Wörter frei aussuchen, das die Mitte einer aufzubauenden Wortkette bildet. Der nächste Spieler kann dann aus den verbliebenen Wörtern eines wählen, das wie ein Dominostein zur bestehenden Wortkette passt, d.h. entweder muss der letzte Buchstabe des neuen Worts mit dem ersten Buchstabe der Kette übereinstimmen oder der erste Buchstabe des neuen Worts mit dem letzten Buchstaben der Kette. Einmal eingesetzte Wörter dürfen nicht ein weiteres Mal verwendet werden. Wem es nicht gelingt, ein Wort an die Kette anzufügen, scheidet aus. Der nächste Spieler beginnt dann mit einer neuen Kette, wobei nur die bislang ungenutzten Worte verwendet werden dürfen. Der letzte verbleibende Spieler gewinnt.

Beispiel: Alice, Bob und Carol spielen. Alice beginnt mit „Idaho“, Bob ergänzt mit „Hawaii“, Carol mit „Utah“, Alice mit „Oregon“, Bob mit „New Hampshire“. Dann entsteht die Kette „Utah“, „Hawaii“, „Idaho“, „Oregon“, „New Hampshire“, die sich nicht mehr fortsetzen lässt, da es weder Bundesstaaten gibt, die mit „u“ enden, noch mit „E“ beginnen. Carol muss also ausscheiden. Danach beginnt Alice eine neue Kette mit „New Jersey“, worauf Bob „Michigan“ nennt, was Alice nicht ergänzen kann. Entsprechend scheidet Alice aus und Bob gewinnt die Partie. David Silverman stellte das Rätsel, ob Alice als beginnende Spielerin einen Sieg hätte erzwingen können und wie sie dann hätte beginnen müssen.

Um das Spiel durchzuführen bzw. es zu analysieren, erscheint eine dynamische Datenstruktur für Wortmengen recht nützlich, die auf Wortanfänge und Wortenden spezialisiert

ist. Im Rahmen dieser Übungsaufgabe ist eine solche Klasse zu entwickeln, die abgesehen von *std*::*string* keine Container-Klassen der STL verwenden und sich nicht auf festdimensionierte Datenstrukturen beschränken darf, d.h. auch andere größere Wortmengen kommen in Frage. Die Klasse muss vollständig ausgestattet sein, so dass u.a. auch ein Default-Konstruktor, ein Kopierkonstruktor, ein Destruktor und ein Zuweisungs-Operator zur Verfügung stehen. Zu unterstützen ist im weiteren

- das Hinzufügen und Entfernen von Wörtern,
- die Abfrage, ob ein Wort in der Menge enthalten ist,
- die Abfrage, wieviel Worte in der Menge enthalten sind und
- das Abrufen von allen Worten, die mit einem bestimmten Buchstaben beginnen oder enden.

Optional können auch der übernehmende Konstruktor (*move constructor*) bzw. die entsprechende Zuweisungs-Operator implementiert werden.

Auf der Vorlesungsseite finden Sie ein kleines Archiv namens *words.tar.gz*, das folgende Dateien enthält:

- *TestIt.cpp*: ein kleines Testprogramm
- *Word.hpp* und *Word.cpp*: eine kleine Klasse für Wörter
- *WordSet.hpp*: eine denkbare Schnittstelle für *WordSet*
- *words.txt*: Textdatei mit den 50 US-Bundesstaaten

Sie können die vorgegebenen Dateien in dem Umfang übernehmen, wie Sie es wünschen. Im einfachsten Falle brauchen Sie nur *WordSet.cpp* zu implementieren. Aber Sie haben die Freiheit, auch die anderen Dateien zu ändern, wenn Ihnen das sinnvoll erscheint. In der Vorgabe ist die private Datenstruktur bei *WordSet* mit **struct** *Words*∗ *words* angegeben. Wenn es Ihnen bequemer erscheint, können Sie natürlich dies direkt durch die privaten Datenstrukturen Ihrer Wahl ersetzen. Alternativ können Sie die privaten Datenstrukturen auch ganz privat und damit aus der Header-Datei herauslassen wie bei dem Vorlesungsbeispiel *ListOfFriends*.

Sie können Ihre Lösung wieder mit *submit* einreichen:

```
thales$ submit cpp 7 WordSet.cpp
```

Wenn Sie die anderen Dateien verändert haben, sollten Sie diese ebenfalls einreichen.

Viel Erfolg!